REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*: 2/11/2013

2. REPORT TYPE: Final Report

3. DATES COVERED *(From - To)*: 3/1/2009 to 11/30/2012

4. TITLE AND SUBTITLE

Detecting Hidden Communications Protocols

5a. CONTRACT NUMBER

5b. GRANT NUMBER: FA9550-09-1-0173

5c. PROGRAM ELEMENT NUMBER

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

6. AUTHOR(S)

Richard R Brooks

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Clemson University
PO Box 340915
Clemson, SC 29634-0915

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research/RSL
875 North Randolph Street
Suite 325, Room 3112
Arlington, VA 22203-1768
Dr. Robert Herklotz

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S): AFRL-OSR-VA-TR-2013-0084

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution A: Approved for Public Release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The work funded by the grant is structured in three parts: We analyzed the vulnerability of the current generation anonymity tools to traffic analysis attacks. We specifically concentrate on SSH security and The Onion Router (Tor) anonymity tools. Our analysis used deterministic hidden Markov models (HMMs). We used traffic timing data to analyze one of the most sophisticated and popular types of cybercrime tools – botnet. We presented two botnet detection methods: centralized botnet traffic detection using HMMs and probabilistic context-free grammars (PCFGs) for centralized and P2P botnet traffic detection. Finally, a hybrid network security scheme that combines the advantages of widely deployed security technologies (intrusion detection systems (IDS) and honeypots) was proposed. The scheduling problem of the security system was modeled as an average decentralized partially observable Markov decision process (DEC-POMDP) and solved using our nonlinear programming (NLP)-based solution method.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
U	U	U	U		**19b. TELEPHONE NUMBER** *(Include area code)*

Reset

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

AFOSR Final Report

Richard R. Brooks

February 11, 2013

Chapter 1

Introduction

This document is the final report for the Detecting Hidden Communications Protocols AFOSR grant with R. R. Brooks form Clemson University as PI. The work funded by the grant is structured in three parts:

I. We analyzed the vulnerability of the current generation anonymity tools to traffic analysis attacks. We specifically concentrate on SSH security and The Onion Router (Tor) anonymity tools. Our analysis used deterministic hidden Markov models (HMMs).

II. We used traffic timing data to analyze one of the most sophisticated and popular types of cybercrime tools – botnet. We presented two botnet detection methods: centralized botnet traffic detection using HMMs and probabilistic context-free grammars (PCFGs) for centralized and P2P botnet traffic detection.

III. Finally, a hybrid network security scheme that combines the advantages of widely deployed security technologies (intrusion detection systems (IDS) and honeypots) was proposed. The scheduling problem of the security system was modeled as an average decentralized partially observable Markov decision process (DEC-POMDP) and solved using our nonlinear programming (NLP)-based solution method.

Four demonstrations were implemented. The first analyzed timing patterns in SSH data streams. We were able to establish sets of discrete time intervals that we could use to infer HMMs of network traffic patterns. Using our HMM statistical hypothesis extensions, we were able to reliably determine the language being typed interactively. Our HMM approach was shown to be able to learn models of network protocols tunneled through encrypted pipes and identify when the protocol is being used.

The second demonstration used timing patterns to break Tor anonymity. Inter-packet delays were collected and used to infer HMMs. We applied a model confidence test [33] to check if the inferred model is representative of the underlying process. By restricting the communication protocol model to the subset of the HMM that was statistically significant, we were able to infer a HMM that was a faithful representation of the underlying protocol. If we monitored two points on the network used by the same Tor session, we were able to identify that they used the same protocol. If we further traced the protocol's path through the Markov model at the two points, it becomes clear that the two points are instances of the same protocol.

In the third demonstration, we used our HMM inference and detection approaches for centralized botnet traffic detection. In this application, we also applied traffic timing analysis, because inter-packet timings of botnets relate to underlying botnet behaviors. Therefore, we used HMMs for automated network traffic analysis to detect C&C communications of centralized botnets. We focused on Zeus botnets (Zbots), one of the largest HTTP-based botnets. Once HMMs were inferred, they were used to detect the botnet communications traffic. Using the HMMs confidence intervals approach [6], we detected whether or not a sequence of traffic data is botnet traffic. Experimental results on traffic data collected on real-world botnets show that this approach can dependably distinguish botnet traffic from normal traffic.

To avoid the single-point failure disadvantage of the centralized structure, hierarchical botnet uses P2P techniques [1,2,22]. In the last demonstration, we attempted to apply stochastic grammars to P2P hierarchical botnet traffic detection. We first constructed hierarchical P2P botnet using Clemson campus computer cloud (Palmetto cloud). Using the HMM detection approach, we found that HMMs can not detect recursive patterns in simulated P2P botnet traffic timings. Therefore, we applied (probabilistic context-free grammars) PCFGs detection approach. The experiment results show that PCFGs have accurate detection rates.

Chapter 2

Demonstrations

2.1 Timing Side Channel Analysis for Language Detection in Secure Shell (SSH) Tunneled Session

If interactive SSH is used, then timing analysis can determine which language is being used. To illustrate this, we set up an experiment where text was transmitted through an SSH tunnel with the sequence of inter-keystroke delays following statistics collected during previous research [15, 16]. We then used the zero-knowledge HMM inference algorithm [26] to construct a HMM consistent with the language structure. The inter-keystroke time delays for the two languages differ due to a variety of factors including, but not limited to:

- keyboard layout
- character/ key-pair frequencies due to language
- muscle memory
- respective grammars

The constructed HMM was used to identify if English and/or Italian is present in text sent through the encrypted pipe. When the text was transmitted, timing data is monitored. These values corresponded to HMM transitions, which were used to compute steady-state probabilities. This allowed us to determine how well the text timing statistics fit the language patterns, in this case, Italian/English. The degree of similarity was measured using confidence intervals [6]. This process is detailed in Figure 2.1.

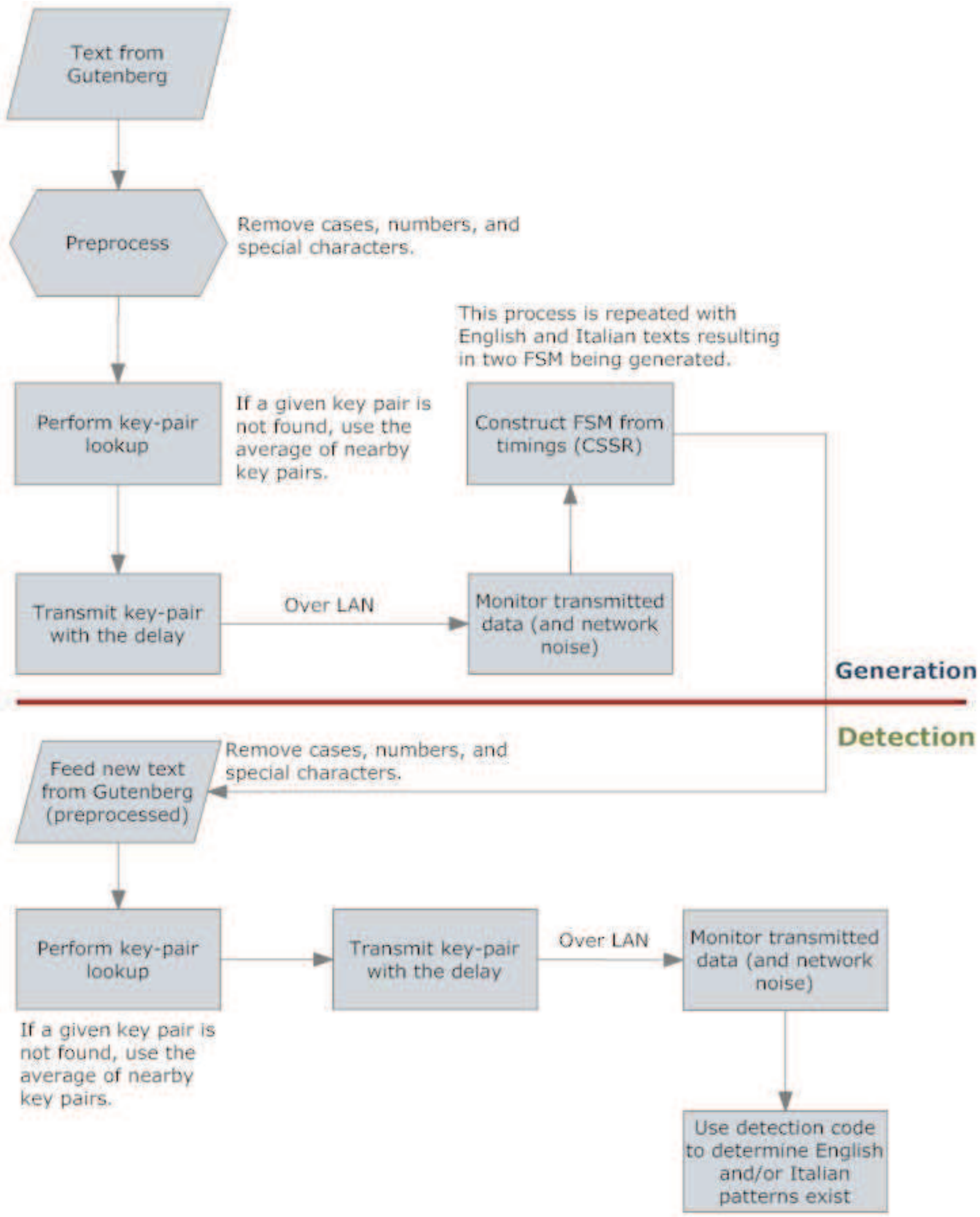


Figure 2.1: Language Data-Flow

2.1.1 Zero-Knowledge HMM Inference

Traditionally, the Baum-Welch algorithm is used to infer the state transition matrix of a Markov model and symbol output probabilities associated with the states, given an initial Markov model and a sequence of symbolic output values [23]. To construct a HMM without *a priori* structural information, we use the causal-state splitting reconstruction (CSSR) algorithm [27, 28, 31], which derives the HMM state structure

and transition matrix from available data samples. CSSR finds statistically significant groupings of the training data that correspond to HMM states. This is accomplished by analyzing the conditional next symbol probabilities for a data window that slides over the training data. This data window increases gradually from a size of two to an *a priori* known maximum window size L. The state structure of the HMM is inferred from the symbol groupings of length L by adding those states to the model that lower system entropy [29, 30].

Except for the training data, the only initial information required to construct the HMM using CSSR is the parameter L, which expresses the maximum number of symbols that are statistically relevant to the next symbol in the sequence. We extend CSSR so that we determine parameter L with no prior knowledge and therefore derive minimum entropy HMMs with no *a priori* information.

To find the correct string length L, we check to see that the HMM inferred using CSSR with string length L is consistent with the model structure inferred using string length L. We verify the consistency of the models by seeing if their interpretation of the symbolic dataset output χ by the process being analyzed is consistent. If we use the notation G_L to represent the HMM inferred with parameter L. Starting from $L = 2$, we calculate the mappings of states to symbols for G_L and those of G_{L+1} respectively. For each state in G_L paired with each state in G_{L+1}, we determine how many times they agreed on the mapping of symbols to states. Since there is no clear start state, we keep the largest value m_L, which measures how similar state machines G_L and G_{L+1} are. If two HMMs assign the symbols in the training data to the same states, then their interpretations are identical. The process repeats with increasing values of L producing a new machine and value m_L with each iteration. As L increases, the CSSR algorithm monotonically improves the ability of the HMM G_L to explain the training data. As the HMMs asymptotically approach the true structure of the process that produced the data, the amount of agreement between G_L and G_{L+1} increases. When the correct value of L is found, there is no new information to be gained by using $L+1$. At this point, the mapping of symbols to states will remain stable (i.e. $mL = m_{L+1} = m_{L+2} = \cdots$) and the process can terminate.

The resulted algorithm is referred to as zero-knowledge HMM inference algorithm and detailed in [26]. In contrast to other techniques, our CSSR extensions create a minimum entropy model without any prior knowledge of the underlying state structure. We tested our results using models where we are certain of the value of L which is necessary for finding the true state structure of the input process.

2.1.2 English and Italian Detection

Our language structure HMMs were inferred from key-stroke data [15, 16] collected from native speakers of English and Italian using their native keyboards. We extracted the keystroke dynamics of each language. If there was no value for any key-pair, the neighbor list of the destination key was consulted. Since the overlaps are too large to effectively distinguish between key-pairs, a clustering approach [11] was used to find distinct classes of key-pairs. Our clustering approach gave us 10 distinct key-pair clusters for English. Italian only had 4 clusters. We used the approach from [33] to determine both the significance of the models and the volume of data necessary for creating a significant model. For the Italian data, a reconstruction with a string length $L = 3$ was possible. We could only use $L = 1$ for the English data because creating a significant model for $L = 1$ would have required a training set of over 11 million key-pairs. Our training sets had approximately 1.1 million key-pairs.

Training data for HMM construction were collected from Project Gutenberg. Recent, (1900 or later), texts were taken and preprocessed to remove case and special characters. The zero-knowledge approach from [26] was used to extract HMMs from the training set. The resulting HMMs are shown in Figures 2.2 and 2.3.

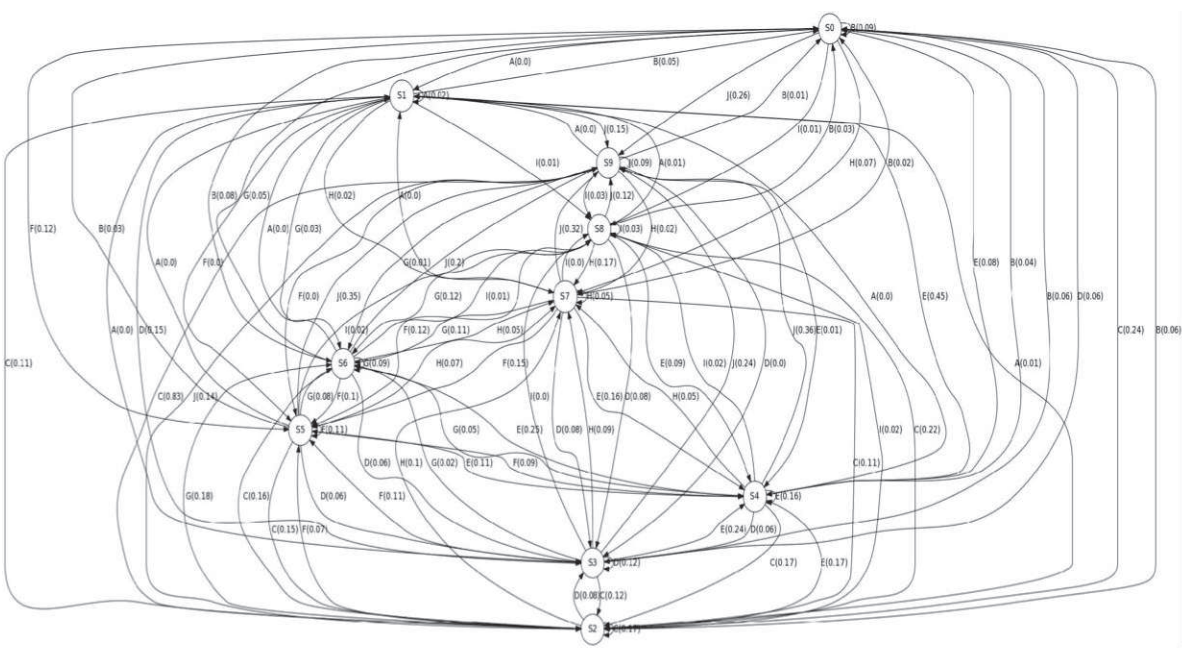


Figure 2.2: English HMM (10 states, 100 transitions)

Using window-size calculations from [24], we found the minimum string length needed to differentiate between the two models, with a 95% true-positive rate, was 77 symbols [25]. A set of 800 English and Italian

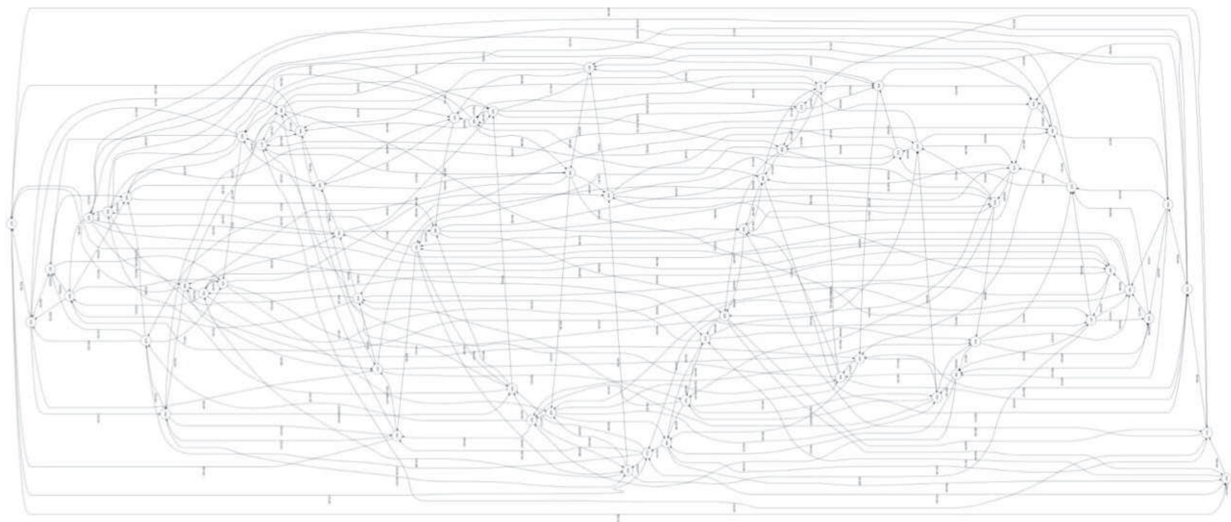


Figure 2.3: Italian HMM (64 states, 253 transitions)

windows were chosen to use for testing. We used the test data to determine the ability of the English and Italian models to detect the language being used in interactive SSH sessions. The testing data was sent through interactive SSH v2 connections. The detection routine used our English and Italian HMMs with the confidence interval (CI) detection criteria [6]. The detection results are shown in Figure 2.4, 2.5, 2.6 and 2.7. In conclusion, the test was successful: using a threshold of 0.0% with the Italian HMM and 89.0% with the English HMM it is possible to detect the presence of either language in a given sample string. That is, if the CI analysis shows that more than 89.0% of the behavior of the English HMM is exhibited by the string, it is English with a 5% false positive rate. This detection was performed in real-time and can be done from a third node as well as the packet contents were not needed, merely the delays between them.

2.2 Detecting Protocols Tunneled through Tor

In this work, we applied statistical methods in pattern recognition to test the privacy capabilities of a very popular anonymity tool known as The Onion Routing network (Tor) [9]. Since Tor is a low-latency system, we applied timing analysis on Tor traffic.

2.2.1 Experiment Setup

A private Tor network shown in Figure 2.8 was implemented on standalone network. Custom client and server programs were set up to communicate through this network. The client would connect to the server

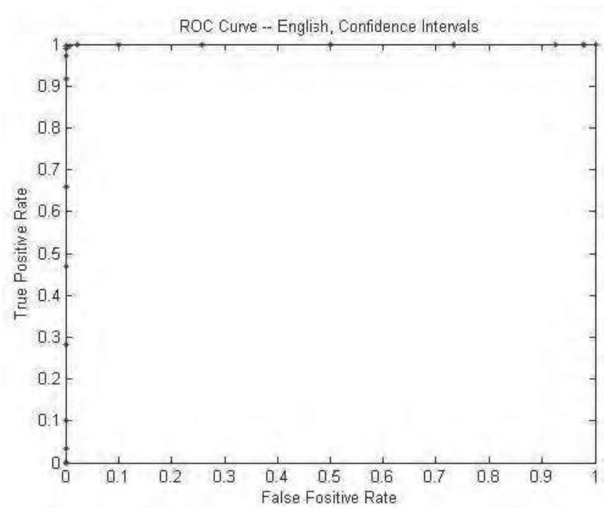


Figure 2.4: English ROC – 95% CI

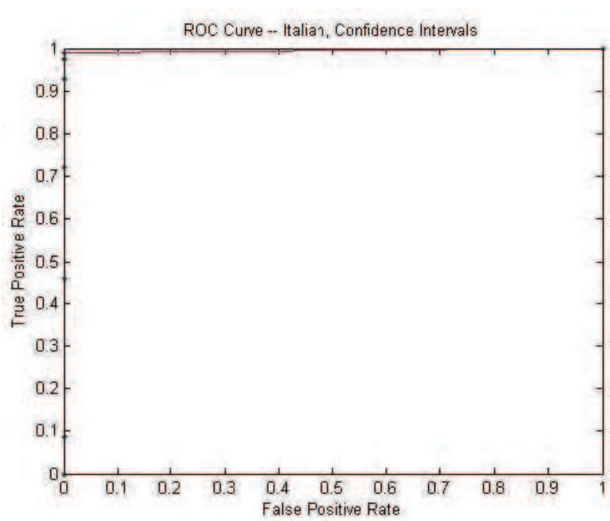


Figure 2.5: Italian ROC – 95% CI

and just listen. The server, once the client had connected, would send data packets to the client based on a preloaded HMM. The model used by the server is depicted in Figure 2.9. The server randomly selected a starting state. To send each packet, a transition was taken from the current state. If there was more than one possible outgoing transition, the transition was chosen randomly, weighted on the probability of each transition. Most importantly, the time between sending each packet depends on the symbol of the transition. Each symbol was assigned a specified time delay in milliseconds and the server waited that amount of time before sending the packet to the client.

We used the zero-knowledge HMM inference algorithm [26] to infer the HMM the server uses by collecting

Thresh.	True Pos	False Pos	True Neg	False Neg	Distance
0.00	401	401	0	0	1.000
	[Repeated 79 times]				
0.80	401	401	0	0	1.000
0.81	401	392	9	0	0.978
0.82	401	371	30	0	0.925
0.83	401	294	107	0	0.733
0.84	401	201	200	0	0.501
0.85	401	103	298	0	0.257
0.86	401	40	361	0	0.100
0.87	401	9	392	0	0.022
0.88	399	3	398	2	0.009
0.89	***399***	***0***	***401***	***2***	***0.005***
0.90	397	0	401	4	0.010
0.91	390	0	401	11	0.027
0.92	367	0	401	34	0.085
0.93	367	0	401	34	0.085
0.94	264	0	401	137	0.342
0.95	188	0	401	213	0.531
0.96	113	0	401	288	0.718
0.97	41	0	401	360	0.898
0.98	14	0	401	387	0.965
0.99	1	0	401	400	0.998
1.00	0	0	401	401	1.000

Figure 2.6: English ROC – 95% CI – Statistics

Thresh.	True Pos	False Pos	True Neg	False Neg	Distance
0.00	397	0	401	4	0.009975
	[Repeated 93 times]				
0.94	397	0	401	4	0.009975
0.95	381	0	401	20	0.049875
0.96	354	0	401	47	0.117207
0.97	245	0	401	156	0.389027
0.98	117	0	401	284	0.708229
0.99	14	0	401	387	0.965087
1.00	0	0	401	401	1

Figure 2.7: Italian ROC – 95% CI – Statistics

the inter-packet timings on the client. The inter-packet time delays, preserved by Tor, were symbolized into ranges and used to construct the models. Using the zero-knowledge HMM inference [26] algorithm, we created HMMs of network processes tunneled through Tor. We followed the process described by the flowchart in Figure 2.10 to reconstruct the HMM.

Figure 2.8: Private Tor Network

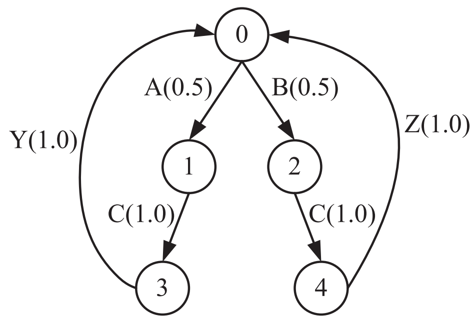


Figure 2.9: HMM Representation of the communication protocol

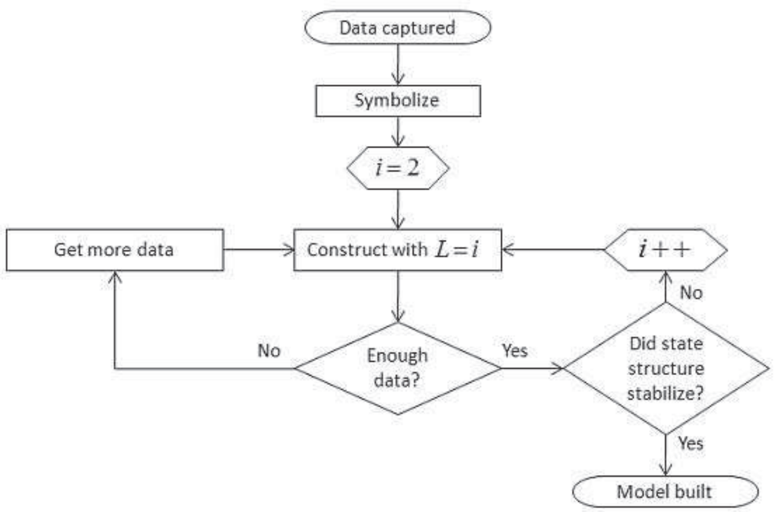


Figure 2.10: Flowchart summarizing the model construction process

2.2.2 Model Reconstruction

Figure 2.11 shows the HMM constructed for the first 200,000 packets. The model confidence test [33] was then run on the model and showed that the required amount of data kept increasing with each set. This was due to the noise (circuit failures and packet drops, etc) of the Tor connection. These glitches caused the packet to arrive later than it should have. All of these events were very low probability and caused the confidence test to increase the amount of data required.

To handle rare transitions and maintain model confidence, we used a threshold on asymptotic state probabilities to prune them out [9]. Following the pruning process, the model in Figure 2.12 results with a significance level of 0.01 (or 1%). By equivalent transformation, it turned out that, the model in Figure 2.12 is essentially the same as the original model in Figure 2.9. As a result, we have successfully inferred the client the protocol used by the server by eavesdropping on the client.

2.2.3 Protocol Detection

After the construction of training models, detection is performed using confidence intervals approach for HMMs [6]. New test data can be fed through a model to determine the intervals and estimate how well the data matches the model. If a match is found, the state sequence, or path, can be used to uniquely describe the data with respect to the model. By comparing these paths, Tor users can be identified. Packet data from any two computers using the Tor network can be matched to a model and their state sequences can be compared to give a statistical likelihood that the two systems are actually communicating together over Tor. We perform experiments on our private Tor network to validate this. Results shows that communicating systems could be identified with a 95% accuracy in our test scenario.

Our detection procedures can be use to break Tor anonymity by matching the timing data of network streams of two or more systems. It is unique from more traditional approaches because they do not rely on maximum likelihood to select the best match out of a set. We only need test data and a single model to determine a statistical measure of how well that model represents the data. For path matching in the Tor attack, we only need two systems both using Tor to tell if they are communicating together or with another party. The adversary does not need to be a global observer. The attack can also be performed in real-time provided that a matching model had already been constructed.

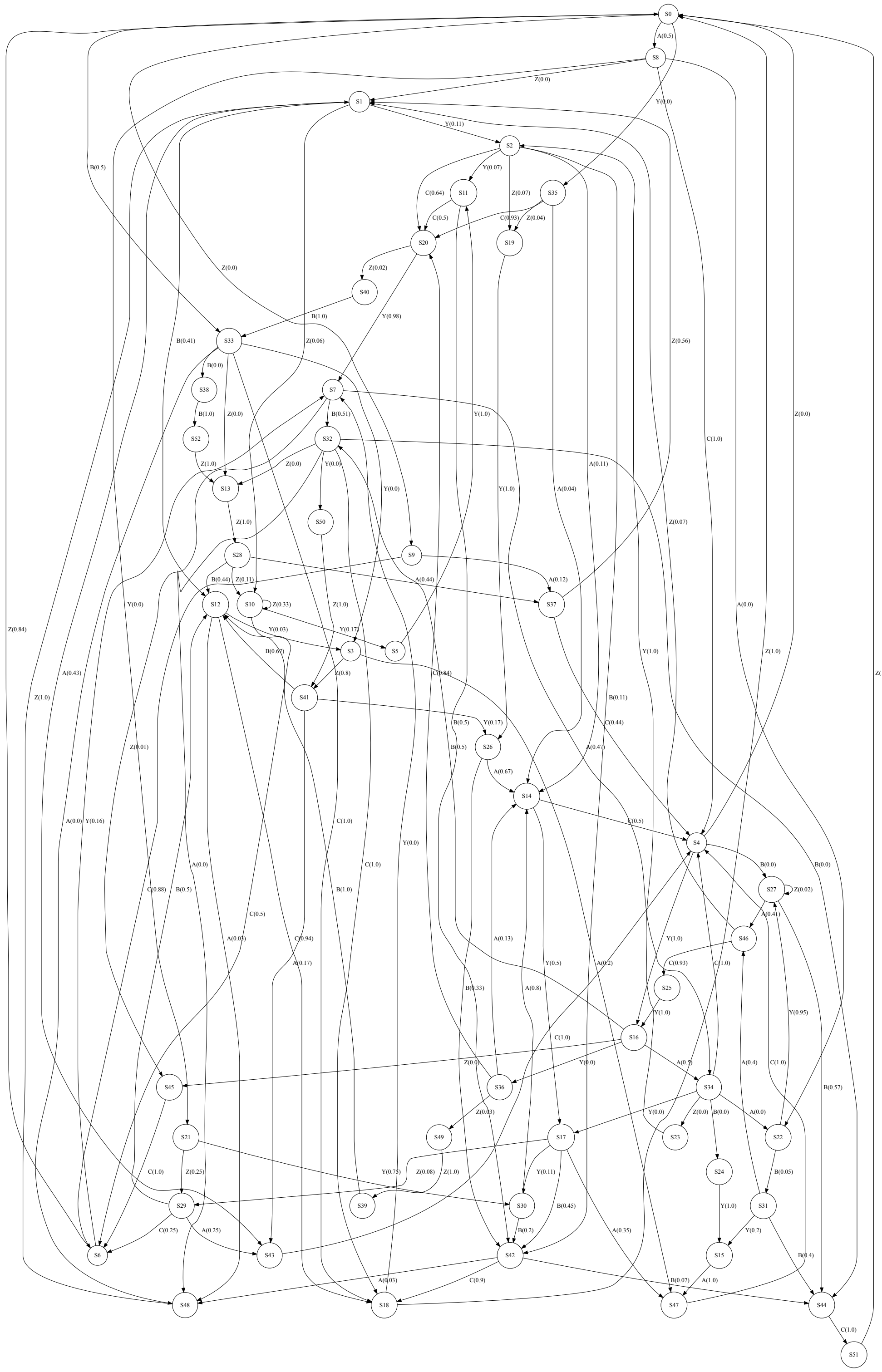


Figure 2.11: Model that is reconstructed from first 200,000 packets of captured data

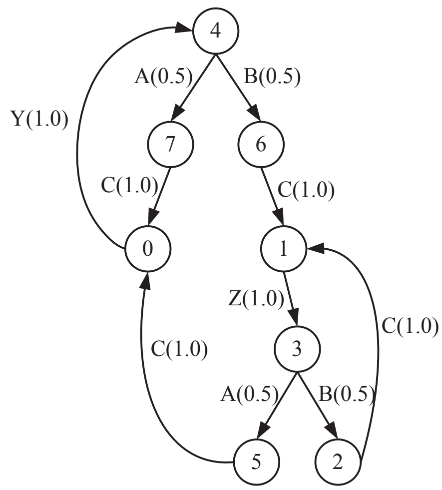


Figure 2.12: Result after pruning low-probability states and transitions

2.2.4 Piercing the Cloud And Marking the Onion: Removing the Anonymity of the Tor Network

In this work, we extend previous work done on removing anonymity from Tor. We explore previous techniques for removing Tor's anonymity developed on a private Tor network, and attempt to reproduce these results on the global public network [3].

We first tried to replicate previous work on the global Tor network. We design multiple paths through the network, so as to prevent any observer from being able to recognize a user based on the time delays between packets. The idea is that multiple paths distort the timing data so as to stop the attacker from recognizing the target. From experiment result, we find that this design is unable to be carried over to the public network. This is mainly due to the level of jitter on the public network overwhelming our earlier method for compromising Tor's anonymity. This method also fails to be able to analyze the noise differences between the different paths in the two and three path tests.

We develop a new method for compromising Tor's anonymity by using a clustering algorithm to analyze the data that we gathered via a side channel timing attack. Since the noise from the global Tor network overrides any observable results, we use neural network as a clustering method to find different symbols in the timing data. It finds data clusters that can be recognized despite the jitter in the global network. We then use the recognizable timing patterns to build HMMs. Using these models we are able to recognize network traffic patterns and reduce Tor's anonymity.

We then establish the multiple paths through global Tor to prevent our side channel attack. Because

the paths don't contain the same nodes, the packet delays are different. This successfully counters our side channel attack and restores Tor anonymity.

2.3 Centralized Botnet Detection

2.3.1 Zbot Traffic

We used our zero-knowledge HMM inference and detection approaches for centralized botnet traffic detection. We focus on Zeus botnets (Zbots) [2], one of the largest HTTP-based botnets. Every Zbot has a centralized C&C server and the communication between the C&C server and bots uses HTTP. Figure 2.13 shows the communication between the bot and the C&C server.

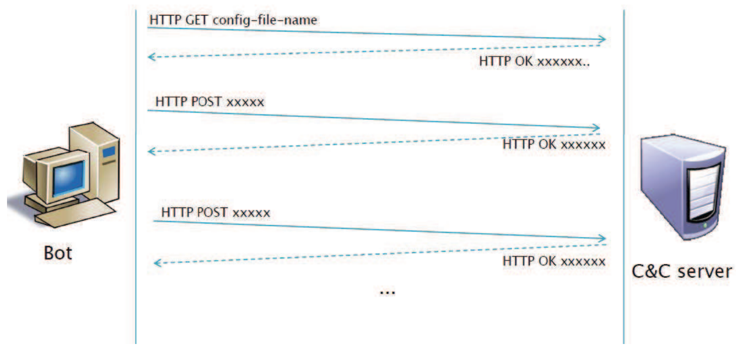


Figure 2.13: Botnet C&C communication

In this application, we also applied traffic timing analysis. Inter-packet delays of botnets relate to command execution time, idling time, contact period and other botnet activities, and therefore are a consequence of botnet behaviors. Sudden changes in communication delays may affect inter-packet timings, however, by checking model significance during HMM inference we remove the influence of this noise [**?**, 24]. Model significance also ensures enough data is used to incorporate complete behaviors of bots. Furthermore, different bots of the same botnet will have similar communication patterns. Therefore, we use HMMs to deduce patterns in inter-packet delays, and apply inferred models to detect similar botnet traffic.

2.3.2 Zbot Detection

We used HMM approaches to detect Zbot traffic. The detection process is described in Figure 2.14. We set up a standalone Zeus botnet to collect traffic data [20]. With the collected training data, we inferred the HMM in Figure 2.15. This HMM is statistically significant according to the model confidence test [**?**].

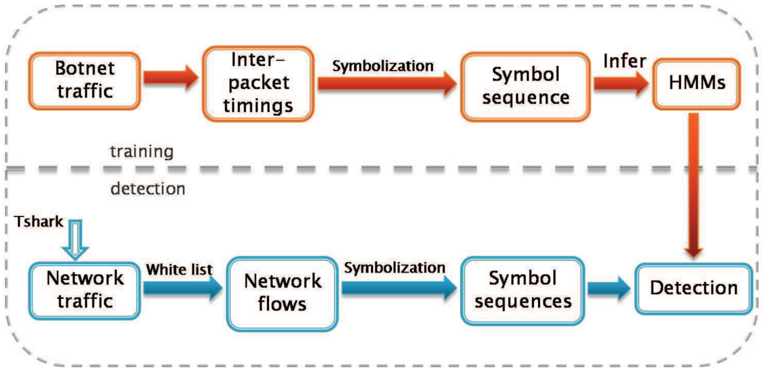


Figure 2.14: Botnet traffic detection

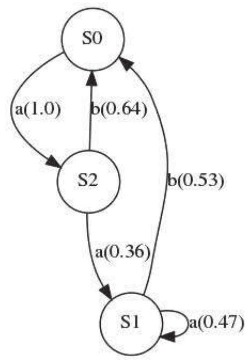
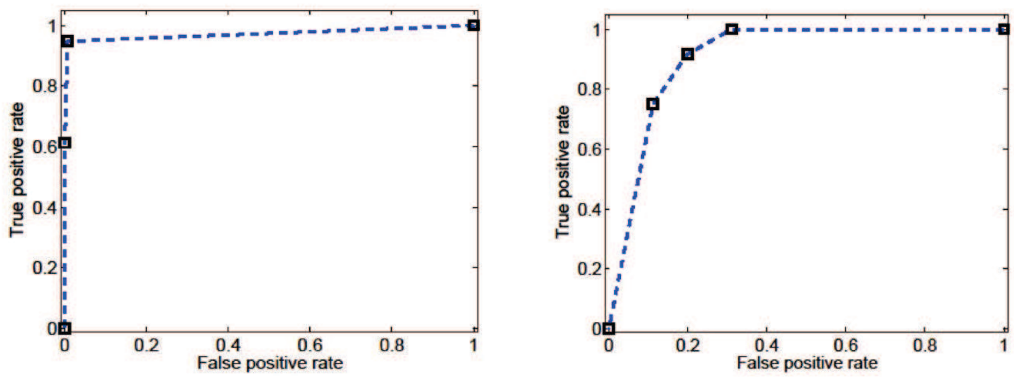


Figure 2.15: HMM of the wild Zbot

Both real-world normal and testing botnet traffic data were then sent to the botnet HMM for detection. Using the confidence interval approach, we obtain the detection ROC curve in 2.16a. With the optimal threshold ([0, 0.667]), our approach gave 94:7% true positive rate and a 0.7% false positive rate. We also compare our results with the autocorrelation approach in [14], which detects botnet traffic based on spatial-temporal correlation of timing data in a local network. While their work checks the periodicity of the traffic data, our approach checks underlying behaviors of the botnet. As shown in Figure 2.16b, our approach is more accurate in detection.

2.4 P2P Botnet Traffic Detection

To avoid the single-point failure disadvantage of the centralized structure, hierarchical botnet uses P2P techniques, in which a C&C server is set up to control a tree-structured computer network. We applied PCFG to P2P hierarchical botnet traffic detection. We first tested our approach with two simulated hierarchical botnets [20] and got the detection result in Figure 2.17. With a high true positive rate and a low false



(a) (b)

Figure 2.16: Detection ROC curves: (a) Zbot HMM detection ROC curve (95% CI); (b) Autocorrelation detection ROC curve

positive rate, we can dependably detect hierarchical botnet traffic using our detection approach.

Data Set Source	PCFG1	PCFG2
TP rate	96.7%	93.4%
FP rate	1.6%	1.6%

Figure 2.17: Simulated botnets detection results

We then applied PCFG to detect real-world P2P botnet traffic. We got a trace of traffic from Storm botnet [13], one of the known P2P botnets. At one point, it accounted for 8% of all malware on Windows systems. In a Storm botnet, the communications between bots may have similar or equivalent probability distributions for malicious actions. Based on this analysis, we developed productions that explains Storm botnet traffic patterns. We also used inter-packet timing delays as traffic patterns for Storm botnet. For P2P botnets, inter-packet timings relate to malicious action timings, publicizing periods and other botnet activity characteristics. In our application, since Storm traffic uses P2P, we are only interested in timings of UDP packets. With this CFG, we can build a LALR parser to parse the symbolized traffic data sets. After symbolization, we parse the training data sets to estimate a PCFG. Using the χ^2 test, we detect whether a data set matches the estimated PCFG. If a data set matches the PCFG, we conclude that this data set generated by the Storm botnet. Using the χ^2 critical value as a threshold and varying it, we plot the ROC curve for detection. The curve is shown in Figure 2.18. It has optimal detection rate 100% TP rate and 0% FP rate. Therefore, by using ROC curves, we can dependably distinguish Storm botnet traffic from normal background traffic.

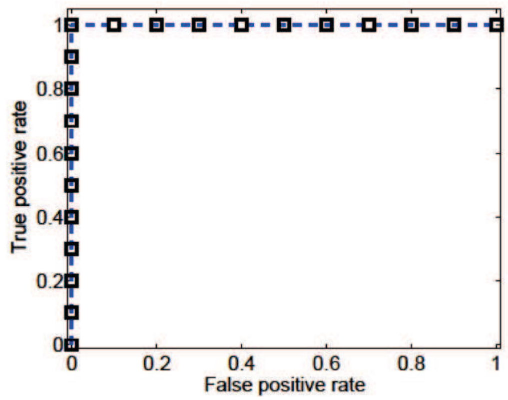


Figure 2.18: Storm botnet traffic detection ROC curve

2.5 Distributed Scheduling for Hybrid Security Scheme Combining IDS and Honeypot

New, flexible and adaptive security schemes are needed to cope with emerging security threats. We proposed a hybrid network security scheme including host-based intrusion detection systems (HIDSs) and honeypots scattered throughout the network. A honeypot is an activity-based network security system, which makes it the logical supplement of the passive detection policies used by IDSs. Since the overhead introduced by the security measures cannot be neglected, device scheduling is needed to balance between security performance and resource consumption. We presented a fully distributed control scheme with the adoption of partially observable Markov decision process (POMDP). This makes our scheme generic and flexible.

2.5.1 System Model

Intrusion detection and system emulation consume a large amount of energy and other resources, including memory, processor usage and disk storage. Thus we need to balance security performance and the resource cost by scheduling IDS and honeypot activities. The scheduling problem was modeled as a discrete-time process. Assume a local area network (LAN) is equipped with $K-H$ HIDSs and H honeypots. Without loss of generality, we also assume the network topology is static. An example network is shown in Figure 2.19.

Suppose each HIDS can operate in three modes: *monitor*, *prevention* and *sleep*. The energy consumption level of these three actions decreases successively. Similarly, the action space of a deployed honeypot can be specified as *monitor*, *analysis* and *sleep*. Further analysis will be carried out if traffic anomalies are detected. Let $S^{(k)}(t)$ denote the state of an arbitrary device k (HIDS or honeypot) at time t, we as-

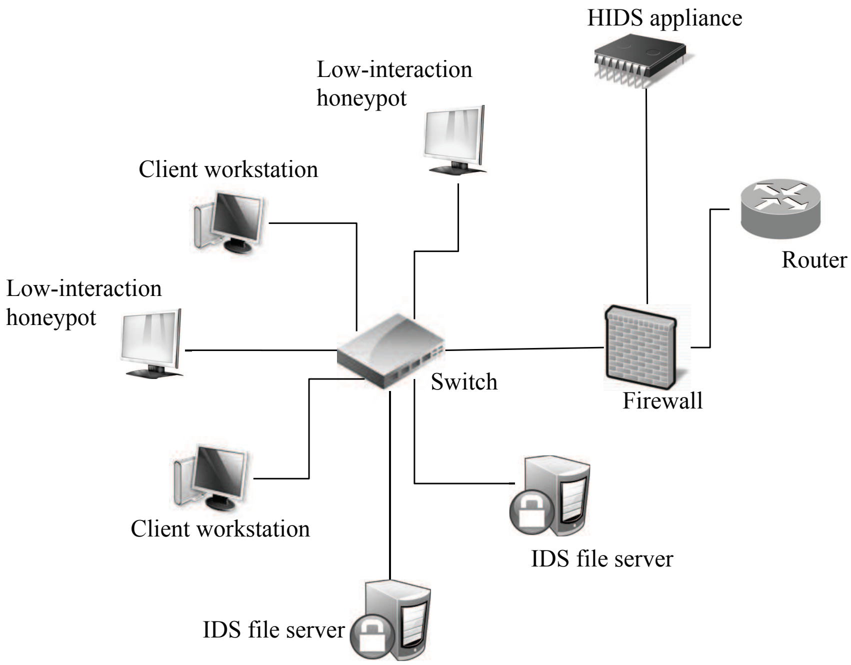


Figure 2.19: Example distributed hybrid security scheme combining HIDSs with honeypots.

sume $S^{(k)}(t) = \langle X^{(i)}(t), Y^{(k)}(t) \rangle$, where $X^{(k)}(t)$ represents the security condition and $Y^{(k)}(t)$ represents the resource consumption level.

Each HIDS only monitors the machine it resides on, ignoring the rest of the network. As a decentralized control scheme, the decision to activate a certain security device is based on its local security condition. An intrusion alarm does not necessarily mean there is an attack, and vice versa. Intrusion detection can make two types of errors: false positive (FP) and false negative (FN). Since the goal of intrusion detection is to precisely differentiate the intrusions from legitimate behaviors, both errors are significant performance indexes of IDSs and have been embodied in the observation probabilities.

In addition, the local state of each host in the network is closely related to the security posture of the entire network. For instance, the activation of a honeypot will positively impact the network's operation and security by distracting adversaries away from the valuable resources in the LAN, and accordingly will mitigate the threat posed to the rest of the network. From the preceding analysis, we selected the decentralized POMDP (DEC-POMDP) to model the scheduling for the distributed system.

2.5.2 NLP-based Solution of the DEC-POMDP

The scheduling model for the hybrid system is modeled as a DEC-POMDP. Thus, we need to augment the POMDP solution method in Equation (3.1) to situations of multiple controllers. The solution of a DEC-POMDP consists of a set of policy graphs, one for each agent. Accordingly, the goal is to optimize a set of finite state controllers (FSC). The formal representation of the NLP-based solution of DEC-POMDPs is:

For variables: $\pi_{\overline{n}s\overline{a}}$ and $g^{(k)}(n_k, {o_k}', {n_k}', {a_k}')$, where $g^{(k)}(n_k, {o_k}', {n_k}', {a_k}') = x_k(n', a')y_k(n, o', n')$

$$\text{maximize } \sum_{\overline{n}} \sum_{s\in\mathcal{S}} \sum_{\overline{a}\in\mathcal{A}} \pi_{\overline{n}s\overline{a}} \cdot r(s, \overline{a}^K)$$

Subject to:

For $\forall s' \in \mathcal{S}, \forall \overline{n} \in \triangle, \forall \overline{a}' \in \mathcal{A}^K$,

$$\pi_{\overline{n}'s'\overline{a}'} = \sum_{\overline{o}\in\mathcal{O}^K} \sum_{s\in\mathcal{S}} \sum_{\overline{a}\in\mathcal{A}^K} \{\pi_{\overline{n}s\overline{a}} \sum_{\overline{o}'\in\mathcal{O}^K} P(s, \overline{a}, s')Q(\overline{a}, s', \overline{o}') \prod_k g^{(i)}(n_k, {o_k}', {n_k}', {a_k}')\},$$

$$\forall n_k \in \mathcal{N}^{(k)}, \forall {o_k}' \in \mathcal{O}, \sum_{{n_k}'} \sum_{{a_k}'\in\mathcal{A}} g^{(k)}(n_k, {o_k}', {n_k}', {a_k}') = 1, \text{for } k = 1, 2, \cdots, K \tag{2.1}$$

The optimal solution to the NLP in (2.1) provides an optimal set of FSCs of the given size. The solution representation owes its availability to one critical factor: each agent behaves independently. That is, all the policy graphs are independent from each other.

Chapter 3

Research Findings

3.1 Inferring Statistically Significant HMMs

We consider the zero-knowledge HMM inference algorith [26] that constructs minimum entropy HMMs directly from a sequence of observations. If an insufficient amount of observation data is used to generate the HMM, the model will not represent the underlying process. Current methods assume that observations completely represent the underlying process. It is often the case that the training data size is not large enough to adequately capture all statistical dependencies in the system. It is therefore important to know the statistical significance level for that the constructed model representing the underlying process, not only the training set.

3.1.1 Model Confidence

The CSSR algorithm [30], as well as its improved algorithm [26], generates deterministic HMMs with each transition uniquely specified by an output symbol, also known as an observation. When these models are dynamically constructed from observations, two questions are raised:

- Does the model match the observations? and
- Are we confident that the model and observations represent the actual underlying process?

The first question refers to the *model fidelity*, which measures the agreement between the inferred model and the training data; the second question refers to the *model confidence*, which means the degree to which a model represents the underlying process that generates the training data. In this work, we address the

second problem. As shown in Figure 3.1, an underlying process is observed over some time interval, creating an ordered sequence of observations. The observations are used to construct a minimum entropy HMM [30]. A lot of research has been devoted to the analysis of model fidelity. Model confidence, in contrast, is a topic previously lacking in other analyses. In the model fidelity literature, the observations are assumed to completely represent the underlying process.

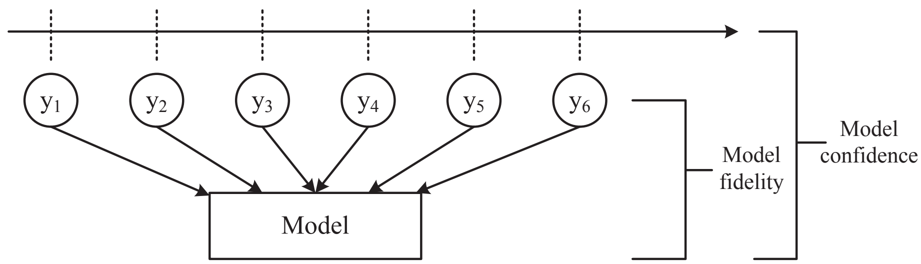


Figure 3.1: Hierarchy of the process, observations, and model showing the relationship between model fidelity and model confidence

3.1.2 Model Confidence Algorithm

Suppose we have constructed a HMM using the zero knowledge HMM inference algorithm [26] from an output sequence. Consider an outgoing transition δ associated with a particular state s. The joint probability of s and δ is ϵ. The analysis of model confidence can be rephrased as to determine if the inferred model will include all transitions with probabilities no smaller than ϵ with desired significance α.

Let π_s be the asymptotic probability of state s, the conditional probability of δ emitting from state s can be calculated with $\gamma_s^\delta = \dfrac{\epsilon}{\pi_s}$. We use z-test to determine if the inferred model includes all transitions leaving s with emission probabilities no smaller than γ_s^δ with desired significance. In the one-tailed z-test, the null hypothesis H_0: $p_s \geq \gamma_s^\delta$ is tested against the alternative hypothesis H_1: $p_s < \gamma_s^\delta$, where p_s represents the conditional probability of an undetected outgoing transition of state s; and accordingly the sample value of p_s is 0. We say more data is needed if H_0 is accepted. Otherwise, we say that sufficient data has been collected.

To use the z-test in this manner, we propose a simple algorithm to perform on-line testing of the observation sequence [33]. The algorithm determines if a constructed model statistically represents a data stream in the process of being collected. We also prove that the z-test only needs to be performed once for the state with minimum asymptotic probability and also finds the minimum amount of training samples needed for δ to be detected with this given level of significance.

In [33], we provide simple illustrations of the concept of model confidence algorithm. The results of these examples illustrate the point that if an insufficient amount of data samples are used, the algorithm only creates a model representative of the data, not of the underlying process. To build a model representing the underlying process, there must be enough data samples available to fully describe that process. The adoption of the model confidence algorithm in the practical examples in Section 2 illustrate the useful application value of the proposed algorithm. In these applications, we used the model confidence algorithm [33] to calculate the amount of training dataset to rebuild the HMM to captures the pattern of interest.

3.2 Similarities between Constructed Markov models

As presented in Chapter 2, we used HMMs to represent patterns of different network traffic. these demonstrations, HMMs were inferred from timing data. Similar data sets can create models that represent the same system with minor differences in probability or structure. It may, however, be difficult to tell if model differences are minor or significant. For detection, we need to find if the inferred model matches the given model. So we developed a metric space for HMMs [19].

3.2.1 Model Equivalence

We define equivalence ($M1 = M2$) as $M1$ and $M2$ accepting the same symbol sequences with statistical significance α. To compare models for statistical equivalence, we extend the confidence interval approach in [6] to use the χ^2 test of equivalence for sets of normal distributions. The procedure is as follows. We generate a sequence of symbols from $M1$ and traverse $M2$ with this sequence. If a path through $M2$ that corresponds the symbol sequence exists, we say that $M2$ accepts this sequence. We then count the number of times each transition being taken. The occurrence probability (joint probability) of each transition is estimated by dividing the corresponding transition frequency by length of the sequence. χ^2-test is used to determine if $M1$ and $M2$ were equivalent at a significance level α. The procedure is in Figure 3.2. Each probability in the smaller model would be inferred using a large number of samples, so the estimated probabilities should be closer to the true value.

3.2.2 Model Distance

When $M1$ and $M2$ are not equivalent, the distance between them is determined by how much their statistics differ with significance α^2. To find this difference, we progressively remove the least likely events from

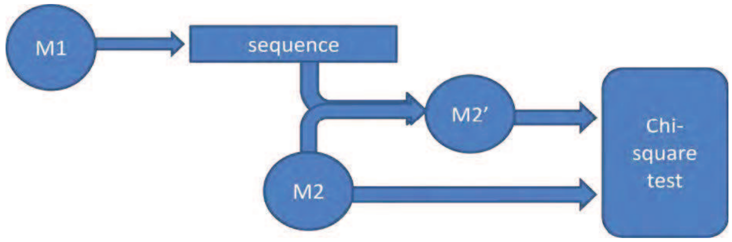


Figure 3.2: Procedure for model equivalence test

both systems until the remaining Markov processes are equivalent. We define a threshold value P_{th} on the joint probability of transitions. Transitions with joint probability not greater than P_{th} are deleted from both models. The resulted models are renormalized and then compared using the approach introduced in Section 3.2.1. The procedures is repeated until two Markov models are tested as equivalent. Therefore, P_{th} iterates from low to high and stops when two models are equivalent. We define the minimum P_{th} that makes $M1$ and $M2$ equivalent as the distance between these two models.

3.2.3 Experiment Results

The proposed model distance approach is used to calculate the distance between $G11$, $G12$, and $G13$ in Figure 3.3. They have similar structures and are only different in the transitions around state $S3$. Event probabilities of transitions leaving state $S3$ (where the differences are located) are in square brackets. The results are shown in Figure 3.4.

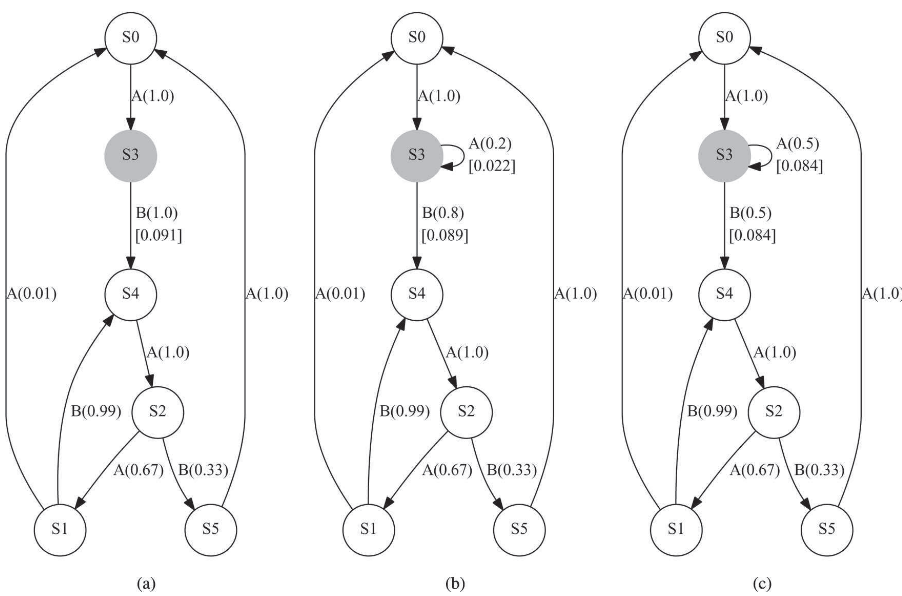


Figure 3.3: Similar models: (a) $G11$; (b) $G12$; (c) $G13$.

Test pair	Distance
$G_{11} - G_{12}$	0.022
$G_{11} - G_{13}$	0.091
$G_{12} - G_{13}$	0.089

Figure 3.4: Model distance results for similar models in Figure 3.3

We also compare our results with those of the KullbackLeibler divergence (KLD) approaches. The results using KLD are shown in Figure 3.5. Since $G11$ cannot accept the sequence generated by $G12$ due to the additional transition in $S3$, $D(G12, G11) = \infty$. However, $D(G11, G12)$ is small since $G12$ can match the sequence generated by $G11$ with a similar probability. It is the same for $G11$ and $G13$. Both cases clearly show that KLD cannot measure the distance for them; however, our approach solves this problem by progressively removing events to a point where the remaining models are equivalent, and it returns a reasonable distance value. This shows a strong point of our approach in which it can measure the distance between all appropriate models while maintaining the triangle inequality.

Test pair	$D(G_i, G_j)$	$D(G_j, G_i)$	$D_s(G_i, G_j)$
$i = 11, j = 12$	0.008	∞	∞
$i = 11, j = 13$	0.028	∞	∞
$i = 12, j = 13$	0.012	0.016	0.014

Figure 3.5: KLD results for similar models in Figure 3.3

3.3 Network analysis by PCFGs

While using HMMs to represent protocols in network, we have one assumption that the protocol has finite state. For context-free grammars (CFGs), they are in a higher level than finite state machines in Chomsky Hierarchy and do not have finite states limitation. Moreover, programming languages can be represented by a CFG, so it is more reasonable to use CFGs to model protocols. We started research with Probabilistic CFGs, where each production rule is augmented with a probability.

3.3.1 Inferring PCFGs

A context-free grammar is a tuple $G = (N, \Sigma, R, S)$, where:

1. N is a set of nonterminal symbols;

2. Σ is a set of terminal symbols;

3. R is a set of production rules in the form of $A \rightarrow \beta$, where $A \in N$ and $\beta \in (\Sigma \cup N)^*$;

4. S is the start symbol in N.

A Probabilistic CFG is a CFG where each production rule has an associated probability, denoted as $p(A \rightarrow \beta)$. For every nonterminal A, we have $\sum_\beta (A \rightarrow \beta) = 1$. The statistical methods we used for HMMs were also applied to PCFGs. We took parsing tools from compiler development and found that we could create LALR (1) (look ahead left-right of 1 character we parsed a data stream 1 character at a time as the data arrived) parse table and used it to detect protocols by analyzing timing side-channel information. The procedure is in Figure 3.6

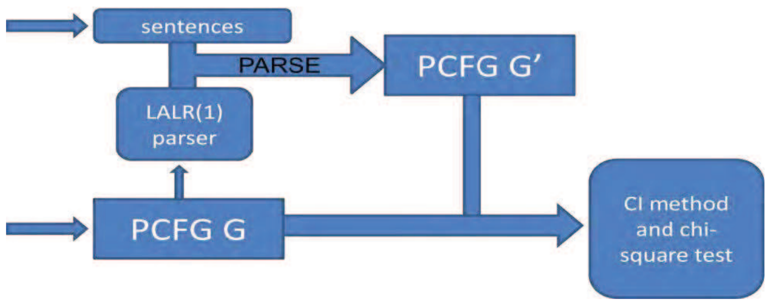


Figure 3.6: Procedure for PCFG detection

We estimated production probabilities while parsing using LALR (1) parse table. Frequency counting method which was similar to the confidence intervals approach was used for each reduce action performed while parsing. The reduce action of parsing using parse table indicated that a reduction with a production rule should be performed for one or more symbols in the sequence. The frequency of the reduce actions of particular production rule in parsing a sentence meant how many times that production rule were used while generating this sentence by the PCFG. The frequency of a particular nonterminal was calculated by summing frequencies of all the production rules whose left hand side is that nonterminal. We estimated the production probabilities by dividing the production frequencies by frequencies of their respective left hand side nonterminals. The confidence intervals of the estimated probabilities are:

$$\left[\hat{p}_{i,j} - Z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{p}_{i,j}(1-\hat{p}_{i,j})}{n_i}}, \hat{p}_{i,j} + Z_{\frac{\alpha}{2}}\sqrt{\frac{\hat{p}_{i,j}(1-\hat{p}_{i,j})}{n_i}}\right]$$

where $\hat{p}_{i,j}$ is the estimated probability for j^{th} production that starts with i^{th} nonterminal, n_i is the frequency of i^{th} nonterminal and $Z_{\frac{\alpha}{2}}$ is from standard normal distribution with significance α. Confidence interval was used for detecting one production, but for the whole PCFG detection, we used χ^2-test. For a set of sentences

and a know PCFG, we built the statistics:

$$\chi^2 = \sum_i \sum_j n_i \frac{(\hat{p}_{i,j} - p_{i,j})^2}{p_{i,j}}$$

where n_i is the frequency of i^{th} nonterminal in the sentence set, $\hat{p}_{i,j}$ is the estimated probability of j^{th} production that starts with i^{th} nonterminal and $p_{i,j}$ is the probability of j^{th} production of i^{th} nonterminal in the known PCFG. If $\chi^2 > \chi^2_\alpha$, we rejected the hypothesis that the sentence set was generated by the known PCFG. The degree of freedom of chi-square test is $|R| - |N|$, where R is the number of productions in PCFG and N is the number of nonterminals.

3.4 Fixed-size Finite-state Controllers for POMDPs with Average Performance Criterion

Partially observable Markov decision processes (POMDPs) are control systems that cannot be observed directly. They commonly model stochastic environments with hidden states. By generalizing Markov decision processes (MDPs) and allowing for more uncertainty, POMDPs provide a more powerful formalism for modelling realistic problems, especially managing systems with noisy data or limited sensitivity. POMDPs have been extensively used in diverse fields [7].

A great deal of research has been devoted to infinite horizon discounted POMDPs. We consider the problem of finding the optimal FSC of a given size for average POMDPs. We transform the POMDP planning problem into a NLP problem. A wide assortment of optimization techniques can efficiently solve large-scale NLP problems [4]. The experiment results in section 3.4.2 show that a small FSC can adequately optimize the selected benchmarks. This suggests that near-optimal approximations may be possible for large-scale problems with reasonable-size FSCs, which indicates that our approach may be able to solve POMDPs whose size is beyond the limits of classical methods.

3.4.1 The Proposed Algorithm

Our approach only requires one input: the size of the FSC ($|\mathcal{N}|$). A nice feature of a POMDP given a FSC is that, the sequence generated by the joint process $\langle N_t, S_t \rangle$ constitutes a Markov chain [21]. A similar conclusion also applies with FSCs stated as follow: Given a POMDP, under any given FSC, the sequence of node-state-action triplets $\langle N_t, S_t, A_t \rangle$ constitutes a Markov chain.

We assume that the optimal Markov chain, i.e., the one with the maximum average reward, is ergodic. That is, the expected reward is constant and does not depend on the initial belief. According to Yu [32], there exists an ϵ-optimal policy. Since any policy can be represented by a FSC [5], this ensures the existence of an ϵ-optimal FSC. Based on the above analysis, we proposed the following NLP-based average POMDP solution in a similar way as the linear programming (LP) approach for solving average MDPs [10].

Theorem 1. *Given an ergodic POMDP, the optimal fixed-size FSC, denoted by $\langle \mathcal{N}, \mathcal{A}, \mathcal{O}, x^*, y^* \rangle$, can be generated from the optimal solutions of the NLP problem:*

$$\boxed{\begin{aligned}
&\mathit{max} \sum_{n \in \mathcal{N}} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi_{nsa} \cdot r(s,a) \\
&\textit{where variables } \pi_{nsa} \textit{ and } g(n,o,n',a') \textit{ are subject to:} \\
&\pi_{n's'a'} = \sum_{n \in \mathcal{N}} \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi_{nsa} \sum_{o' \in \mathcal{O}} T(s,a,s) Z(a,s',o') g(n,o',n',a'), \quad \forall n' \in \mathcal{N}, \forall s' \in \mathcal{S}, \forall a' \in \mathcal{A}, \\
&\sum_{n' \in \mathcal{N}} \sum_{a' \in \mathcal{A}} g(n,o',n',a') = 1, \quad \forall n \in \mathcal{N}, \forall o' \in \mathcal{O}, \\
&g(n,o',n',a') \geqslant 0, \quad \forall n \in \mathcal{N}, \forall o' \in \mathcal{O}, \forall n' \in \mathcal{N}, \forall a' \in \mathcal{A}.
\end{aligned}} \tag{3.1}$$

Let $\mathbf{g}^* = \{g^*(n,o',n',a')\}_{n \in \mathcal{N}, o' \in \mathcal{O}, s' \in \mathcal{S}, a' \in \mathcal{A}}$ *be part of the optimal solution of the NLP in* (3.1)*, we can construct the corresponding optimal internal state transition probabilities* $y^*(n,o',n')$ *by*

$$y^*(n,o',n') = \sum_{a' \in \mathcal{A}} g(n,o',n',a') \tag{3.2}$$

and the optimal action selection probabilities $x^*(n',a')$ *by*

$$x^*(n',a') = \frac{g^*(n,o',n',a')}{y^*(n,o',n')} \tag{3.3}$$

3.4.2 Experiments

We demonstrate the utility of our approach by applying it to a set of frequently used benchmark problems for POMDP algorithm testing. For detailed information, we refer the reader to [8]. We used the sparse nonlinear optimizer (SNOPT) integrated in the optimization software called AIMMS to solve the NLP problems. This solution uses the SQP method [12]. AIMMS is freely available for academic use. All the computation was performed on a Intel Core 2 DUO E8400 3.00 GHz computer.

To illustrate the effectiveness of our approach, a summary of our results, in comparison with the results of two other algorithms can be found in Table 3.1. Perseus PBVI is one of the leading POMDP approximation techniques and has been shown to be very efficient for discrete POMDPs [**?**]. We also compare the performance of our approach with the results published in [32] using the lower approximation scheme (LAS).

Table 3.1: Experimental comparisons of the proposed approach with other algorithms

Problem	**FSC**			**Perseus PBVI**		**LAS**
	$\lvert\mathcal{N}\rvert$	**a. r.**	**time(sec)**	**a.r.**	**time(sec)**	**a.r.**
2s2a2o(max)	2	10.1454	<1	8.5013	11	N/A
Tiger(max)	3	1.0838	<1	-1	13	N/A
DAS(max)	2	0.3569	<1	-9.721	21	N/A
paint(max)	2	0.17	<1	-0.0011	66	-0.172
bridge(min)	2	259	<1	2515	37	241.798
shuttle(max)	2	2.6316	<1	1.1202	224	-1.835
maze(max)	2	4.2413	<30	2.1357	766	N/A
hallway(max)	6	0.043	<100	0.064	22641	N/A
machine maintenance(max)	2	0.095	<30	0.0368	41	N/A
tag(max)	2	0	<30	-1	266	N/A
tiger-grid(max)	2	0	<30	0	2643	N/A

[1] "max" means to maximize the limiting average reward while "min" means to minimize the limiting average cost.

In Table 3.1, the columns with identifier **a.r.** contain the limiting average rewards computed using different algorithms. $|\mathcal{N}|$ is the size of the FSC. The computation time for solving the NLPs are given in the column labeled **time(sec)**. All the experiments using Perseus are configured to iterate a significant number of steps (2000 steps in our context) to ensure that the algorithm converges to a stable value. So we also give the time taken to complete the iteration for each problem.

The experimental results show that in almost all the domains our approach achieves competitive performance through the adoption of very small FSCs. Our NLP-based algorithm often needs a very succinct policy, i.e., a small FSC, to achieve a competitive optimization effect, even for some large-scale problems. In other words, our NLP-based POMDP algorithm allows us to leverage most of what a fixed-size FSC offers in POMDP optimization. Consider, for example, "tiger-grid" problem. While our formulation may not have a distinct advantage in terms of solution quality, the policy representation ($|\mathcal{N}| = 2$) is much more compact than the value function given by Perseus, which contains 20 hyperplanes. Furthermore, it took

the SNOPT solver less than one minute to find the optimal solutions, even for some large-scale problems (maze20, hallway, tag and tiger-grid). Finally, it should also be noted that the results of Perseus PBVI are obtained based the assumption that the initial belief is given, which gives the Perseus PBVI a decided advantage. If the initial belief is inaccurate or unknown, the performance may be worse than the results shown in Table 3.1. But our approach does not require *a priori* knowledge and the computed results are invariant to initial beliefs.

Chapter 4

Summary

Stochastic models are used extensively in science and engineering to explain and predict natural processes. In this work, we investigated the application of stochastic models (HMM, PCFG and POMDP) to network security. In this chapter, we give a summary of our work and achievements.

We proposed pattern detection approaches with stochastic models for traffic timing analysis. We developed inference and detection algorithms for HMMs and PCFG. With these approaches, we inferred the protocols tunneled through SSH and Tor, measured the noise in the Tor communication and detected traffic that are generated from centralized and P2P botnets. From a sequence of observations, we inferred the HMMs without any *a priori* information about the structure and initial transition probabilities [26].

A model confidence test [33] was used to check the significance of the inferred HMM. We find the level of confidence that the model and data represent process under observation. Our method uses the observation data to dynamically determine an upper bound on the probability of an unobserved transition occurring. If this upper bound is not sufficient for the observed situation, we explained how user-defined thresholds can be used to determine new bounds. However, our approach should be free of the problem of overtaining even when this assumption does not hold. Some of the undesired patterns would be excluded from the constructed model due to the threshold ϵ.

To compare the similarity of HMMs for accurate detection, we proposed a normalized statistical metric space [19]. Our Markov metric compares HMM based on underlying system statistics, which is fundamentally different from the graph isomorphism problem from graph theory. Using the χ^2 test, we determine the equivalence of two HMMs. If they are not equivalent, the Markov metric removes states and transitions until the models are equivalent within a given statistical significance. This measures the distance between two

models. Compared to KLD, which is a widely used HMM similarity.

HMMs are probabilistic state machines, which are the simplest models of computation. Therefore we extended the detection approach to use PCFGs, which have more expressive power in pattern representations. We proposed a simple statistical approach for PCFG detection. We estimated production probabilities from either tree sets, or sentence sets with LALR parser. The χ^2-test was used to determine whether a data set matches a given PCFG, or whether two data sets are equivalent. From illustrative examples, our approach has better detection rates compared with the previously used inside-outside algorithm.

Using HMM inference and detection approaches, we presented a centralized botnet traffic detection application [17]. We inferred HMMs from Zbot traffic timing data. The inferred HMM detect botnet traffic using confidence intervals of the state probabilities. Experimental results on real-world network traffic show that this approach appropriately differentiates botnet traffic from normal traffic, and has a higher true positive rate and a lower false positive rate than the autocorrelation analysis.

Hierarchical P2P botnets have arisen to avoid the disadvantages of centralized botnets. HMMs failed to detect recursive and structural patterns in P2P botnet traffic. We used PCFGs to detect P2P botnet traffic [18]. From traffic timing data of a real-world Storm botnet, we estimated PCFGs, which maps probability distributions from traffic data into production probabilities. Detection results show that based on ROC curve analysis on χ^2 statistics, we can appropriately differentiate Storm botnet χ^2 from normal P2P χ^2.

In addition to exploring the vulnerabilities of existing security products, more effective and adaptive measures are needed to tackle new attacks. We presented a distributed security system combining HIDSs and honeypots [34]. Since all devices are placed in the production network, judicious management of available resources impacts the performance of the application. Thus, methods that optimize device scheduling have great importance. The scheduling of the proposed application is formulated as an average DEC-POMDP. This allows each HIDS and honeypot to independently make decisions. In the proposed scheme, the security devices are dynamically selected based on the current security posture and resources.

Bibliography

[1] The threat from p2p botnets. http://www.securelist.com/en/blog/654/Lab_Matters_The_threat_from_P2P_botnets.

[2] Zeus gets more sophisticated using p2p techniques. http://www.abuse.ch/?p=3499.

[3] James Nicholas Ashworth. Piercing the cloud and marking the onion: Removing the anonymity of the tor network. Master's thesis, Clemson University.

[4] D.P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 2004.

[5] D. Braziunas. Pomdp solution methods. Technical report, 2003.

[6] R. R. Brooks, J. M. Schwier, and C. Griffin. Behavior detection using confidence intervals of hidden markov models. *IEEE Trans. on Systems, Man, and Cybernetics, part B*, 39(6):1484 –1492, December 2009.

[7] A. R. Cassandra. A survey of pomdp applications. *AAAI Fall Symposium*, 1998.

[8] A.R. Cassandra. Tony's pomdp file repository page. http://www.cassandra.org/pomdp/examples/index.shtml, 2009.

[9] Ryan Michael Craven. Traffic analysis of anonymity systems. Master's thesis, Clemson University.

[10] J. Filar and L. Vrieze. *Competitive Markov Decision Processes*. Springer-Verlag, New York, NY, USA, 1997.

[11] B. Fritzke. Fast learning with incremental rbf networks. *Neural Processing Letters*, 1(1):2 – 5, 1994.

[12] P.E. Gill, W. Murray, and M.A. Saunders. Users guide for snopt version 7: Software for large-scale nonlinear programming. Technical report, San Diego, CA, June 2008.

[13] G. Gu. *Correlation-Based Botnet Detection In Enterprise Networks*. PhD thesis, College of Computing, Georgia Institute of Technology, 2008.

[14] G. Gu, J. Zhang, and W. Lee. Botsniffer - detecting botnet command and control channels in network traffic. *15th Annual Network and Distributed System Security Symposium*, 2008.

[15] D. Gunetti and C. Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security*, 8(3):312–347, 2005.

[16] K. Hempstalk. *Continuous Typist Verification using Machine Learning*. PhD thesis, University of Waikato.

[17] C. Lu and R. R. Brooks. Botnet traffic detection using hidden markov models. In *Proceedings of the 7th CSIIRW*.

[18] C. Lu and R. R. Brooks. P2p hierarchical botnet traffic detection using hidden markov models. In *Learning from Authoritative Security Experiment Results (LASER) Workshop*, 2012.

[19] C. Lu, J. Schwier, R. Craven, L. Yu, R. Brooks, , and C. Griffin. A normalized statistical metric space for hidden markov models. *IEEE Trans. on Systems, Man, and Cybernetics, part B*, In Press.

[20] Chen Lu. *Network Traffic Analysis Using Stochastic Grammars*. PhD thesis, Clemson University.

[21] N. Meuleau, K.E. Kim, L.P. Kaelbling, and A.R. Cassandra. Solving pomdps by searching the space of finite policies. In *Proc. of UAV*, pages 417–426, Stockholm, Sweden, 1999.

[22] G. Ollmann. Botnet communication topologies. Technical report, 2012. White Paper of Damballa.

[23] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 – 286, 1989.

[24] J. Schwier. *Pattern Recognition for Command and Con-trol Data Systems*. PhD thesis, Clemson University, Clemson, SC, 2009.

[25] J. Schwier, R. R. Brooks, and C. Griffin. Methods to window data to differentiate between markov models. *IEEE Transactions on System Man and Cybernetics, Part B: Cybernetics*, 41(3):650 – 663, 2010.

[26] J. M. Schwier, R. R. Brooks, C. Griffin, and S. Bukkapatnam. Zero knowledge hidden markov model inference. *Pattern Recognition Letters*, 30:1273 – 1280, 2009.

[27] C. Shalizi. *Causal architecture, complexity, and self-organization in time series and cellular automata*. PhD thesis, University of Wisconsin-Madson, 2001.

[28] C. Shalizi and J. Crutchfield. *Journal of Statistical Physics*, (140):819 – 881, 2001.

[29] C. Shalizi and K. Shalizi. Blind construction of optimal nonlinear recursive predictors for discrete sequences. *arXiv:cs.LG/0406011 v1*, June 2004.

[30] C. Shalizi, K. Shalizi, and J. Crutcheld. An algorithm for pattern discovery in time series. *arXiv:cs.LG/0210025 v3*, November 2002.

[31] C. Shalizi, K. Shalizi, and J. Crutcheld. Pattern discovery in time series, part i: Theory, algorithm, analysis, and convergence. Technical report, 2002.

[32] H. Yu and D.P. Bertsekas. On near optimality of the set of finite-state controllers for average cost pomdp. *Mathematics of Operations Research*, 33(1):1 – 11, February 2008.

[33] L. Yu, J. Schwier, , R. Craven, R. R. Brooks, and C. Griffin. Inferring statistically significant hidden markov models. *IEEE Tras. TKDE*, PP, 2012.

[34] Lu Yu. *Stochastic Tools for Network Security: Anonymity Protocol Analysis and Network Intrusion Detection*. PhD thesis, Clemson University.